

Requirements Working Group Information Paper

Presented at the 1996 INCOSE Symposium. Prepared by the Requirements Working Group of the International Council on Systems Engineering, for information purposes only. Not approved by INCOSE Technical Board. Not an official position of INCOSE.

Characteristics of Good Requirements

Pradip Kar
Armament Systems Division
United Defense Limited Partnership
4800 East River Road, MS M239
Minneapolis, Minnesota 55421.

Michelle Bailey
Naval Air Warfare Center, China Lake
Mail code 52C000D
China Lake, California 93555.

Abstract. This paper identifies and documents the major characteristics of well defined requirements for both individual requirements and aggregates of requirements such as those found in specifications. Characteristics exhibited by well defined requirements are described. The descriptions are followed by a brief discussion of the characteristics. A few examples of well defined requirements are provided. Some supporting characteristics of requirements are also described. These supporting characteristics are not properties of the requirements themselves but are attributes assigned to individual requirements to assist in managing the requirements through the life cycle. The paper was written and reviewed by members of the requirements working group of the INCOSE.

INTRODUCTION

It is generally recognized that the development of good requirements is essential to quality product design. However, it is well known that requirements for many programs in the past have been poorly written. The reasons for this shortcoming are many and include a lack of clear definition of user need and insufficient time and effort dedicated to requirements definition. Also there are few examples of well defined requirements for reference.

One of the fundamental problems associated with writing good requirements is that most engineers are not specifically trained to write requirements. The basics of well defined requirements are clarity, conciseness and simplicity. Elegant, entertaining prose is not needed. In the words of Albert Einstein, "When you are out to describe the truth, leave elegance to the tailor". It is clear that to write good requirements, engineers need to develop writing skills which are not taught in school. They also need good and readily accessible examples of both well and poorly written requirements.

This paper describes the essential characteristics of well defined requirements. Characteristics of both individual requirements and aggregates of requirements (as embodied in a specification) are provided. Some examples of well defined requirements are included in each category. This characterization can be used for

requirements development as well as in requirement reviews and audits for assessing the quality of

requirements. It can also be used in Systems Engineering and management process improvements. The paper also describes a few supporting characteristics of requirements. These supporting characteristics, while not essential for writing individual or aggregate requirements, assist in the process of requirements management, specially where the total population of requirements is large.

DEFINITIONS

The definitions for the following terms used in this paper, are from IEEE Std 1220-1994.

Constraint. A limitation or implied requirement that constrains the design solution or implementation of the systems engineering process, is not changeable by the performing activity, and is generally non allocable.

Requirement. A statement identifying a capability, physical characteristic, or quality factor that bounds a product or process need for which a solution will be pursued.

Specification. A document that fully describes a physical element or its interfaces in terms of requirements (functional, performance, constraints and physical characteristics) and the qualification conditions and procedures for each requirement.

System. The top element of the system architecture, specification tree, or system breakdown structure that is comprised of one or more products and associated life cycle processes and their products and services.

CHARACTERISTICS OF REQUIREMENTS

The characteristics of requirements are their principal properties. Characteristics may apply to individual requirements; or to an aggregate of requirements. A well defined, mature set of requirements should exhibit certain individual and aggregate characteristics. These characteristics are listed below, with synonyms in parenthesis. The description of each characteristic is followed by a brief discussion of the characteristic and an example of a requirement displaying the required characteristic.

CHARACTERISTICS OF INDIVIDUAL REQUIREMENTS

The following paragraphs describe and discuss the desired characteristics of good requirements.

Necessary. The stated requirement is an essential capability, physical characteristic, or quality factor of the product or process. If it is removed or deleted, a deficiency will exist, which cannot be fulfilled by other capabilities of the product or process.

Discussion. This is a primary characteristic. In order to be a well defined requirement, the requirement statement must exhibit this characteristic. There is no room in a specification for unnecessary or "nice to have" requirements because they add cost to the product. An example of a necessary requirement for a combat vehicle could be "*The vehicle's combat loaded weight shall not exceed 35 Tons*". The condition "*combat loaded*" is defined elsewhere in the body of the specification. This requirement imposes a major constraint on the design of the vehicle and it is usually based on transportability, utilizing existing resources. If this requirement is deleted from the specification, a major need (i.e. transportability) will not be met, even if the vehicle is capable of satisfying every other requirement. Accordingly, the requirement is necessary.

One good test of necessity is traceability to higher level documentation. In the case of system specifications, traceability may be checked to user documentation such as the Operational Requirements Document (ORD). If there is no parent requirement, the requirement may not be necessary.

Concise (*minimal, understandable*). The requirement statement includes only one requirement stating what must be done and only what must be done, stated simply and clearly. It is easy to read and understand.

Discussion. During requirements definition there are often many arguments as to what exactly is meant by 'one requirement'. The answer in most cases requires engineering judgment. An example of this can be the requirement for an alarm. Within a military system, both a visual and an audible alarm is often required to warn the crew about abnormal or unsafe conditions. Also, the same alarm is used for multiple conditions, such as high temperature, low oil pressure and low fuel level. The question here is what constitutes a single requirement? Should there be a single requirement for the alarm with perhaps a table defining the conditions under which the alarm is activated? An example of a requirement following this approach can be *"The element shall provide a visual and audible alarm under all conditions listed in Table 3-10. The alarm shall be activated no longer than 1 second after the condition exists."* Is this one requirement? Or should we write a requirement for each condition? Should the visual and audible parts be separated into two requirements? On the one hand we do not want to write multiple requirements in one paragraph; on the other we do not want the number of requirements to proliferate without reason. Each requirement will have to be managed and verified; and has a cost associated with it.

One approach to making a decision on this question is to determine how the requirement will be verified. Obviously one would like to verify that both the visual and audible alarms work together. So any test or demonstration to verify the requirement must include both. Therefore, the visual and audible parts should be combined into a single requirement. Further, if the conditions providing inputs to the alarm can be incorporated into the same test or demonstration, these should also be included in the same requirement. The conclusion is: this is a requirement for a single alarm, which can be verified using a single test or demonstration. Accordingly, it is a single requirement.

To be concise, the requirement statements must not contain any explanations, rationale, definitions or descriptions of system use. The place for these texts are analysis and trade study reports, operational concept documents or user manuals. A link can be maintained between the requirement text and the supporting analyses and trade studies in a requirements data base so that if desired, the rationale and explanations can be obtained rapidly.

Implementation free. The requirement states what is required, not how the requirement should be met. A requirement statement should not reflect a design or implementation nor should it describe an operation. However, the treatment of interface requirements is generally an exception.

Discussion. This characteristic of a requirement is perhaps the hardest to judge and implement. What exactly is meant by implementation free? At the system level, requirements can be truly abstract or implementation free. An example of a requirement for a shipboard system at the system level is: *"The system shall be capable of engaging sea skimming anti ship cruise missile (ASCM) targets."* This sentence should be followed by expected performance data (a quantification of what "engaging" means) against specific ASCM targets. It can be seen from the requirement statement that no specific implementation has been stated, so the system designer is free to pursue alternative, competing system designs, all capable of fulfilling the requirement.

The system requirements have to be implemented by a system design solution. After the system designer has performed a trade study between alternatives and selected a candidate solution, the system requirement stated in the previous paragraph, have to be allocated to the elements defined by the system design. This incremental procedure of allocating requirements to the next lower level elements, dependent on system design, has led to the remark: "one level of design is the requirement at the next lower level." In the ASCM example above, let us say that the system design solution consists of a radar element to detect and track the targets; a command and control element to perform target recognition, identification, and weapon assignment and control functions; and a missile to engage and destroy the target. Then there will be three elements below the system level on the system specification tree and the system requirement will have to be allocated to these three elements. The element requirements will be for a radar, a command and control and a missile. The allocated requirement for the radar element can be: *"In a clear environment, the radar element shall be capable of detecting 0.1 (meter)² ASCM targets at ranges of up to 20 KM with a probability of detection of no less than 0.9 and probability of false alarm of no greater than 10^{-6} ."* Note that the requirement statement reflects the design at the system level (the solution is a radar and associated command and control; and missile) but implies no specific radar design. This allows the radar designer to pursue alternative radar designs. The conclusion is that a requirement is implementation free at the level that it is being specified, but is a result of the design activity at the level above it.

Interface requirements are usually an exception to the implementation free rule. Interface requirements are specified in interface control drawings or ICDs which describe a specific design of an interface or mating parts. For example in the case of an electronic interface, the ICD may include part numbers of the interfacing connector(s); pin assignments for various signals; waveform and timing information; and source and sink impedances on each line. In this case the requirement must provide complete information so that the two sides of the interface can be designed to work as specified when connected to each other. Interface requirements are usually specified as a result of an interface control working group (ICWG) activity. The ICWG is usually chaired by the system integrator and has members from contractors representing both sides of a specific interface. The ICWG establishes both the data required to go across the interface and the design implementation of the interface.

Attainable. (*achievable or feasible*). The stated requirement can be achieved by one or more developed system concepts at a definable cost. This implies that at least a high level conceptual design has been completed and cost tradeoff studies have been conducted.

Discussion. This characteristic is a test of practicality of the numerical value or values set forth in a requirement. It signifies that adequate analyses, studies and trades have been done to show that requirement can be met by one or more concepts and that the technology costs associated with the concept(s) are reasonable within the cost constraints of the program.

For example, consider the following requirement for a radar element: *"In a clear environment, the radar element shall be capable of detecting 0.1 (meter)² ASCM targets at ranges of up to 20 KM with a probability of detection of no less than 0.9 and probability of false alarm of no greater than 10^{-6} ."* It is well known that radars operating in the microwave frequencies are more or less horizon limited due to straight line propagation of electromagnetic waves at these frequencies. Under the circumstances, required detection ranges which are much beyond the radar horizon, (the 20 KM range is within the capability of a mast mounted shipboard radar) should be viewed with suspicion unless it is specified under anomalous propagation conditions.

Complete. (*standalone*) The stated requirement is complete and does not need further amplification. The stated requirement will provide sufficient capability.

Discussion. Requirements should be stated simply, using complete sentences. Each requirement paragraph should state everything that needs to be stated on the topic and the requirement should be capable of standing alone when separated from other requirements. Many people seem to have trouble specifying a system parameter called growth completely. Let us examine a basic statement: *"The vehicle shall permit growth"* as an initial idea in trying to specify growth. The question is how much growth and in which areas? Remember, the requirement should be complete and not need amplification elsewhere. Growth is usually provided in new design systems for two reasons:

- Pre Planned Product Improvements (P³I) and
- Through life growth.

Early P³I analysis may indicate the need to fit new equipment into a vehicle, which will need additional space, power, cabling, cooling and weight margins. Through life growth is system growth due to the addition of new capabilities in a system to address new requirements which may be imposed during the life cycle of the system but which were not known during system development. An example of this type of growth is the addition of a whole new mission such as tactical ballistic missile defense (TBMD) to the Aegis combat system on U.S. Navy ships. Other types of through life growth are small changes to fix problems and provide minor capability improvements in response to user need.

Margins must be provided for both types of growth, although it is easier for P³I growth. A requirement statement, for example, in the area of weight growth, may then be stated as follows: *"The vehicle shall permit 12% weight growth."* This requirement, addresses both how much and in which area, and will permit the vehicle designers to design the chassis, engine/transmission and the suspension subsystems taking into account the growth margin.

Consistent. The stated requirement does not contradict other requirements. It is not a duplicate of another requirement. The same term is used for the same item in all requirements.

Discussion. This characteristic of well defined requirements is usually well understood and does not cause much discussion. However, in a large set of requirements which are not well organized by some clearly defined categories, it may be hard to spot duplications and inconsistencies. It is therefore important to organize the requirements (say in accordance with MIL STD 490A) so inconsistencies can be spotted during reviews and audits. It is also important to maintain a glossary of terms being used on the program because the meaning of some words are domain dependent.

Unambiguous. Each requirement must have one and only one interpretation. Language used in the statement must not leave a doubt in the reader's mind as to the intended descriptive or numeric value.

Discussion. This is an area where a formal requirements language for systems engineering may ultimately help. The English language can be unstructured and in some cases the same sentence may mean different things to different people. What is needed are some standard constructs and some words to avoid. While the following list is incomplete, it will point the reader in the right direction.

Standard Constructs. One important word that is included in most DOD specification requirements is the word "shall" for all requirements. It is a statement of imperative need and indicates that the requirement must be verified. A standard construct for a requirement is therefore: *"The system(or element or equipment) shall ..."*. Other requirements are stated as a goal in specifications. The goal requirements are not imperatives, and sentences containing them do not contain the word "shall". An example of a goal requirement is: *"The goal is to provide a maximum range capability of 45 Kilometers."*

The word "will" is not used to signify a requirement. It may be used in a sentence containing a statement of fact such as *"vehicle tests will be conducted at government test facilities"*. They may also be used to indicate that some events will take place in the future. *"Test instrumentation will be provided by the government in 1998."*

When a requirement is defined by referencing another standard, use a construct such as "in accordance with" or "as specified in". An example requirement is *"System parts and equipment requiring identification shall be marked in accordance with MIL STD 130."*

Do not use "Minimum" and "Maximum" to state limits. Use "No less than" or "No greater than". This standard construct avoids the ambiguity associated with the limiting values. The requirement *"Gun alignment error in elevation shall be no greater than 1.5 milliradians"* is not vague because it is clear that a 1.5 milliradian error is permissible. On the other hand, if the requirement was stated as *"Gun alignment error in elevation shall be 1.5 milliradians maximum"*, there would be some ambiguity as to whether the limiting value i.e. 1.5 milliradians was permissible. This rule does not mean that the words "minimum" and "maximum" cannot be used at all - just do not use them to state limits. For example a requirement can be stated as *"The gun system's maximum range shall be no less than 25 KM"* to convey that the maximum range of the gun system must be equal to or greater than 25 KM.

Words to avoid. Specific words that should be avoided because they are vague and general are "flexible", "fault tolerant", "high fidelity", "adaptable", "rapid or fast", "adequate", "user friendly", "support", "maximize" and "minimize" (*"The system design shall be flexible and fault tolerant"* is an example). Other words that should be avoided are "and/or", "etc." and "may".

Verifiable. The stated requirement is not vague or general but is quantified in a manner that can be verified by one of these 4 alternative methods: inspection, analysis, demonstration or test.

Discussion. Verifiability is an important characteristic of a requirement. The verifiability of a requirement should be considered at the same time that a requirement is being defined. A requirement which is not verifiable is a problem, both for the customer and the contractor because it involves the acceptability of a system. In order to be verifiable, the requirement should be stated in measurable terms such as: *"The overall length of the system shall be 105+/-0.5 inches"* (verifiable by inspection) or *"Gun alignment error in elevation shall be no greater than 1.5 milliradians"*(verifiable by test).

Questions are often asked as to what exactly is the difference between a demonstration and a test; and how to decide between a demonstration and a test, when planning a verification strategy. The answer to the first question is extent of instrumentation. A test is fully instrumented while a demonstration involves observation and simple recording of information such as time. A well known demonstration which is required on many programs is a maintainability demonstration. The objective of a maintainability demonstration is to verify the maintainability of a system, usually specified as the mean time to repair (MTTR).

During the demo a trained soldier or sailor (made available by the user community) attempts to repair a series of artificially induced faults on the system, using the technical documentation and available spare parts. For each fault, the only measurements are time taken to diagnose the problem, correct the problem and re- test the system for correct operation. The achieved MTTR is computed from the measured times.

A test, on the other hand, utilizes a large number of instruments. A shipborne missile firing test to determine missile performance against a target utilizes both shipboard instrumentation and missile borne telemetry to determine not only the end results but intermediate results as well to determine whether the

missile performed to requirements throughout the flight.

As can be expected, testing is the most expensive and also the most reliable method of verification. Verifying every requirement by test can be expensive. When determining a verification strategy, trade studies may have to be performed to decide the optimum strategy. This is where analysis comes in. Analysis is often much cheaper than test and can be used effectively in a supporting role. For example, computer models of the system can be used to verify system performance under certain (hard to achieve in real life) conditions, especially after the model has been validated against test results.

CHARACTERISTICS OF AGGREGATE REQUIREMENTS

Aggregate requirements are a set of requirements for a system or element that specify its characteristics in totality. Usually these aggregates are found in specifications. Characteristics of individual requirements are applicable to aggregates also. The following characteristics are applicable to aggregates.

Complete. The set of requirements is complete and does not need further amplification. The set of requirements has addressed all categories (see supporting characteristics), and covers all allocations from higher levels.

Discussion. This characteristic addresses the difficult problem of identifying requirements which are necessary but are missing from the set. There are several approaches to identifying missing requirements. One is to develop the system operations concept and an associated set of scenarios. Walking through these from cradle to grave: how will the system be delivered? how will it be deployed? how will it be used, upgraded and disposed of, help to identify missing requirements. The next step is to walk through the same set of scenarios and play the "what if" game. What if something goes wrong? This step usually uncovers a whole new set of requirements. Another approach is to develop a checklist of topics or areas such as a specification outline as given in MIL STD 490A or DOD STD 2167A and verify that requirements exist in each topic area or if they do not, there are good reasons for it. Yet another is to check the aggregate against a higher level specification (if one exists) to verify that all allocated requirements have been included in the set. Often we leave out the obvious such as *"The ship shall float."*

Consistent. The set of requirements does not have individual requirements which are contradictory. Requirements are not duplicated. The same term is used for the same item in all requirements.

Discussion. This characteristic addresses the problem of identifying unnecessary or conflicting requirements which are inadvertently included in the set. Assignment of a project unique identification to each requirement, and thorough reviews should eliminate these requirements.

SUPPORTING CHARACTERISTICS

These are secondary properties or attributes of individual requirements which provide supplementary information about the requirement, its relationship to other requirements and source documents; and assist in requirements management. These supporting characteristics are not essential in all cases and if an automated requirement management tool is used, the tool may generate some or all of these attributes automatically.

Project unique identification. (PUID). The PUID is a unique identifier assigned to a single

requirement for identification and tracking. It may be either numeric or alpha numeric. The PUID assists in identification, maintaining change history, and provides traceability.

Assigning a PUID is probably a better method to track and maintain requirements on an individual basis than a title or some such textual description. PUIDs can be purely numeric or alphanumeric. An alphanumeric PUID, in addition to being a unique identifier, can provide information about the requirement. For example, the PUID SYS0001 clearly shows that the requirement is a system requirement. On the other hand, the numeric PUID can be automatically assigned to newly formed requirements by many requirement management tools.

Level. This attribute indicates the level at which the specific requirement is applicable in the System Hierarchy. For example, level I may indicate a top or system level requirement. Level II may be the segment level requirement. Requirements written at the right level will provide adequate information without constraining the designer's freedom. While not an essential characteristic, levels can be used to provide statistics on requirements fan out and verification levels.

Category. Categories are used to classify requirements. There are two major categories:

- **Program requirements.** These are customer or user requirements imposed on contractors through contractual vehicles other than specifications. Examples of documents in which program requirements are listed are the contract itself and the contract statement of work (SOW). Program requirements include: compliance with federal, state or local laws including environmental laws; administrative requirements such as security; customer/contractor relationship requirements such as directives to use government facilities for specific types of work such as test; and specific work directives (such as those included in Statements of Work and Contract Data Requirements Lists). Program requirements may also be imposed on a program by Corporate policy or practice.

Program requirements are different from the requirements that have been discussed so far. These are not requirements imposed on the system or product to be delivered, but on the process to be followed by the contractor. Program requirements should be necessary, concise, attainable, complete, consistent and unambiguous. Program requirements are managed in the same manner as product requirements.

- **Technical (product) requirements.** These are requirements applicable to the product or service to be delivered. Product requirements are usually listed in specifications or interface control drawings. These are further classified by these requirement types:

a) Functional. Qualitative requirements describing what the system needs to do without describing them in quantitative terms. These requirements are usually descriptive and are verified by the summation of the associated performance requirements. An example of a functional requirement for a radar element is: *"The radar element shall be capable of detecting targets."*

b) Performance. These are quantitative requirements of system performance, and are verifiable individually. One performance requirement associated with the functional requirement example in the previous paragraph can be *"In a clear environment, the radar element shall be capable of detecting 0.1 (meter)² ASCM targets at ranges of up to 20 KM with a probability of detection of no less than 0.9 and probability of false alarm of no greater than 10⁻⁶."* In fact an appropriate paragraph heading for this requirement would be *"Target Detection"*, the function with which it is associated. There are usually several performance requirements associated with a single functional requirement.

c) Design constraints. These requirements identify the constraints under which the system is required to

operate or exist. Size and weight limitations are included in this category, as are environmental requirements. An example of a design constraint in the area of weight is "*The vehicle's combat loaded weight shall not exceed 35 Tons*".

d) Interface. Interface requirements are the definition of how the system is required to interact with external systems (external interface), or how subsystems within the system interact with each other (internal interface). Interface requirements are often an exception to the 'implementation free' rule of requirements.

e) Non requirement. In a strict sense this is not a requirement type. Very often some explanatory text is provided in source documents which are not requirements, but which is maintained in the requirements data base for completeness and to re- create the original source document, if needed.

Compliance level. There are two levels of compliance:

- **Mandatory.** This attribute is applicable to all requirements defined in sentences containing the word "shall". These requirements must be verified.
- **Goal or objective.** These indicate that this level of performance is desired by the customer. They are not mandatory but need rationale and documentation if they are not met.

Allocated to. Each requirement must be allocated to one or more lower level components. This can be done by allocating the requirement to a component name or to the name of the specification that will define the component requirements. Allocation provides insurance that all requirements are flowed down to the next level. It is a basis for checking that lower level requirements are responsive to the higher level requirement.

Parent PUID. This attribute provides a link to the immediate parent of a derived or lower level requirement. For a lower level requirement, this PUID will represent the next higher level requirement. This attribute is required to provide vertical traceability through the specification tree of the system.

Source. The top level document, supplied by the user, that is the sources for all program requirements. Each requirement is traced to this document and to a unique paragraph number. The document can be a specification, operational requirements document or statement of work. This attribute is required to provide direct traceability for individual requirements to user documents.

Specification number, specification title, paragraph number and paragraph title. These attributes are important when searching and printing requirements. They are used to associate requirements with the appropriate document.

Rationale. This attribute provides a link to the data or tradeoff analyses which support the requirement. The supporting data may include the reason or reasons a requirement is needed; any assumptions made at the time the requirement was formulated; numbers and locations of analysis or trade study reports which support formulation of the requirement and its current value or values. In a large system the rationale for every requirement is not readily available. This attribute provides an index to the rationale for every requirement.

Verification method. This attribute describes and tracks the selected verification method for the requirement. The alternatives are: inspection, analysis, demonstration and test.

Verification documents. This attribute provides a link to the number, title and paragraph number of the

verification Plan and Procedure (such as a test plan) used to verify the requirement.

Change notices. This attribute records the number and date of specification change notices affecting the stated requirement. It assists in maintaining the change history associated with this requirement.

Risk. This attribute provides a record of the risk associated with a requirement, if any. A quantitative assessment of the risk and the date of the assessment is provided.

TPM Parameter. This attribute provides a record of the Technical Performance Measure Parameter in which the requirement is included (The requirement itself may be a TPM parameter) and the current value of the parameter.

Current status. This attribute provides a record of the current status of the requirement. The alternatives are:

- TBD (to be defined) - this indicates that the value of the requirement has not been defined yet. It is good practice to use TBDs for requirements which do not have defined values. It can be used to provide metrics such as the number of requirements which are not yet defined.
- TBR (to be reviewed)- this indicates that a preliminary value is available but needs further review.
- Defined. - This indicates that a final value for the requirement has been obtained through analysis and trades.
- Approved. - The requirement has been reviewed and approved by the appropriate authorities.
- Verified. - The requirement has been verified in accordance with the verification plan.
- Deleted. - The requirement is no longer applicable to the program.

Standards. The standards or conditions under which a requirement must be met.

SUMMARY

Writing good requirements is difficult, requires careful thinking and analysis, but is not magical. Requirements which have the characteristics described in this paper will be easier to meet and help insure that the customer gets what he wants. Time spent up front, carefully defining and articulating the requirements is the first step towards insuring a high quality product.

ACKNOWLEDGEMENTS

The authors wish to thank Ivy Hooks (Compliance Automation/Vital Link), Beth Simon (McDonnell Douglas Aerospace), Dave Jones (Texas Instruments); and Allan Ray, John Bangs and Saumya Sanyal (United Defense) for reviewing the paper and providing useful suggestions and comments.

BIBLIOGRAPHY

1. Writing Good Requirements, Ivy Hooks: Proceedings of the Fourth Annual Symposium, NCOSE working Group Papers Volume II.
2. An Introduction to Systems Engineering, J. Douglas Sailor.
3. Current System Development Practices using the Hatley/Pirbhai Methods, Derek Hatley, The journal of NCOSE Volume 1 Number 1 July-September 1994.
4. Application of the System Engineering process to define requirements for Computer Based design tools, Benjamin Blanchard, Wolter Fabrycky and Dinesh Verma March 1994.

5. Requirements Management/Traceability: A case Study - NASA's National Launch System, Mary Beth Pinkerton & Frank R. Fogle, Proceedings of the Second Annual Symposium, NCOSE
6. Development & implementation of an integrated Systems Engineering environment, R.M. Harwell, Proceedings of the Second Annual Symposium, NCOSE.
7. Requirements for Development of Software Requirements, Brian W. Mar, Proceedings of the Fourth Annual Symposium, NCOSE.
8. What Is A Requirement? Richard Harwell, Erik Aslaksen, Ivy Hooks, Roy Mengot, Ken Ptack, Proceedings of the Third Annual International Symposium, NCOSE.
9. IEEE Std 1220-1994 (Issued February 1995 for trial use).

ABOUT THE AUTHORS

Pradip Kar is the Systems Engineering Manager for the Armament Systems Division of United Defense Limited Partnership, a partnership of FMC and Harsco corporations. Over the past 14 years he has worked on a number of engineering development programs such as the Army's Crusader and Bradley Fighting Vehicles, and the Navy's Aegis and Mk 41 Vertical Launching System programs. Prior to working at United Defense, he was an officer in the Indian Navy and spent time developing a command and control system. He has been a member of INCOSE since 1992. He has a B.S degree in Electrical Engineering from London University and has done graduate work in Computer science at the University of Minnesota.

Michelle Bailey is currently a member of the Range Architecture Office, Naval Air Warfare Center Weapons Division, China Lake, Ca. which has responsibility for major investments within the Navy's West Coast test ranges. Previous experience includes leading the software development effort on the AIM-9R air to air missile and subsequently serving as the systems engineer for that effort. Other experience includes working on advanced signal processing systems and a one year tour in the Pentagon where she wrote the Naval Science and Technology Requirements. Other publications include the China Lake design process handbook, the China Lake System Engineer Handbook, "System Engineering for all Engineers" NCOSE 1994 Symposium, and "HWIL Contributions to Navy Test and Evaluations" SPIE 1996 Symposium.